이동단말 관리서비스를 위한 OMA-DM 프로토콜의 임베디드 XML파서 설계

안희준*, 김용호*, 정승호*, 이재광**, 박병주**

*서울산업대학교, **㈜DKI 테크놀러지

Design of an OMA-DM Protocol XML Parser for Embedded Mobile Device

Heejune Ahn*, Yong-ho Kim*, Seung-ho Jeong*, Jaekwang Lee, Byeongjoo Park****

*Seoul National University of Technology, **DKI Technology Inc.

E-mail:heejune@snut.ac.kr

요 약

이동통신 단말기의 기능과 서비스가 증가로 이동 단말관리를 위하여 SyncML과 OMA-DM의 표준이 정의되고 활용되고 있다. 이 두 프로토콜은 XML 기반 메시지방식에 기초하고 있으며, Sequence와 Atomic 등 복합 명령어 구조를 사용하고 있어, 메모리와 계산속도 등에 제약이 있는 임베디드 시스템에서의 효과적인 구현을 필요로 한다. 본 연구는 자체 OMA-DM 에이젼트의 개발 과정에서 설계한 효과적인 임베디드 OMA-DM용 XML 파서의 설계에 대하여 소개하고, 검증서버 및 ㈜ DKI사의 DM서버에서 실험한 결과를 제시한다.

1. 서론

이동통신 단말기는 점점 많은 사용자를 확보하고 있으며, 초기의 음성서비스를 넘어서 멀티미디어, 인터넷 등 다양한 응용을 지원하면서 개인정보디바이스 (personal information device)로 변화되고 있다. 2009년 현재 단말에는 보통 수십개 이상의 응용 프로그램이 존재하며, 각 프로그램의 구성도 복합화되고 있다. 따라서 이동통신 단말의 소프트웨어 시스템의 복잡이 증가하면서, 프로그램 오류와 기능 모듈의 갱신, 새로운 프로그램 및 펌웨어의 설치 등 단말을 최신의 상태로 유지하기 위한 필요성이 증대되었다. 이러한 작업을 개인사용자가 직접하는 것은 번거롭고 시스템 위험한 일이다.

현재 대부분의 PC용 프로그램은 자사의 비표준적인 방식을 사용하여 이러한 문제를 해결하고 있다. OMA-DS/DM[1]은 이러한 문제를 해결하기 위하여 이동 서비스 사업자, 모바일 장치 제조업체, Microsoft를 비롯한 소프트웨어 공급업체 등 200여 개 업체가 모여 만든 국제표준단체인 OMA(Open Mobile Alliance)에서 제정한 표준이다.

2. SyncML & OMA-DM 프로토콜 분석

SyncML(Synchronization Markup Language)과 OMA-DM(OMA Device Management)은 이동통신 단말장치와 컴퓨터 간에 데이터 동기를 제공하고, OMA-DM은 SyncML에 기반하여 단말 관리 (부트스래핑, 고장 점검, 관리, 응용 및 펌웨어 신규설치 및 갱신) 제공한다.

OMA-DM의 프로토콜 구조는 그림 1과 같으며 전통적인 관리자-애이젼트 구조를 사용하고 있다.

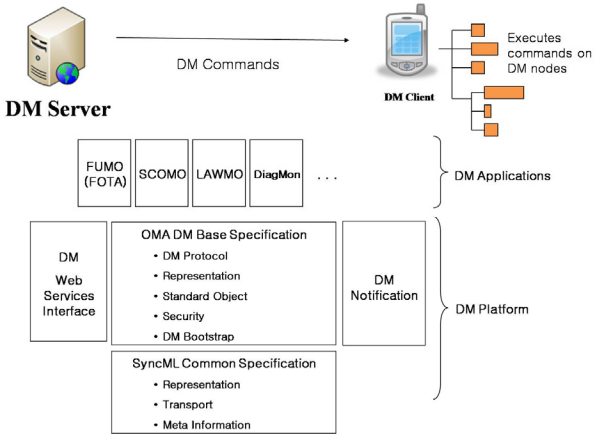

그림 1. OMA-DM의 구조

프로토콜은 세션, 패키지, 메시지의 3단계 구조로 되어있는데, 메시지 단계의 데이터는 하나의 XML(eXtensible Markup Language)문서를 사용한다.

SyncML 과 OMA-DM의 표준안을 토대로 본 XML 구조상의 특징은 다음과 같다.

- 인증과 명령 구분 (SyncHdr, SyncBody)
- 대용량 메시지 전송(Large Object)
- 복합 코맨드 지원(sequence, atomic, sync)
- 메시지당 다중명령 및 참조 방식

이론적으로는 OMA-DM XML은 일반적인 DOM 파서를 통하여 충분히 구현이 가능하다. 그러나 임베디드 상황에서 DOM 파서를 사용하기에는 메모리와 성능에 있어 어려움이 있으며, 경우에 따라서는 DOM 파서가 지원이 안 되는 환경도 존재한다 [2]. 현재 공개 소프트웨어로 확보 가능한 OMA DM 라이브러리인 syncML 툴킷 [3]은 이러한 점에 문제가 있으며, 또한 이후에 살펴 보겠지만 Funambol [3] 역시 완전한 기능을 지원하지 못한다. 최근 2년간 국내에서도 OMA-DM의 구현에 대한 연구가 발표되었으나, 역시 이러한 문제를 가지고 있다[5].

3. 본 구현의 OMA-DM 파서 설계

개발한 에이전트는 DMManager, DMProcessor, DMBuilder 그리고 TransportAgent, 크게 네 가지 부분으로 나뉜다. DMManager는 응용부로 사용자 인터페이스 및 시스템 구성을 관리를 하며, DMProcessor는 패키지와 메시지를 다루는 단계로 DM서버에서 에서 보내온 명령어로 이루어진 메시지를 받아 시스템의 DM Tree 노드의 값을 변경한다. DMBuilder는 메시지의 파싱과 포맷팅을 담당하는 단계로 DMProcessor에서 처리한 결과에 대하여 메시지를 생성한다. 실제 전송은 Transport계층에서 OBEX, WAP, HTTP를 사용한다. 지면의 제약상 타 구현과 크게 상이하지 않은 부분은 제외하고, DMProcessor에서의 XML 처리리를 효과적으로 하기 위한 구현에 대하여 설명한다.

메시지는 명령들로 구성되는데 (status도 일종의 명령임) 다음 표 1과 같으며, OMA-DS(Data Synchronization)에서 쓰이는 Sync명령이나 Search명령은 본 연구에서 다루는 OMA-DM에서는 쓰이지 않으므로, 처리하지 않도록 변경하여 OMA-DM에 맞도록 설계하였다.

표1. OMA-DM에서 사용되는 명령어

명령어	전달자	설 명
Add	Server	노드를 생성
Atomic (복합)	Server	Atomic에 종속된 명령어들을 하나의 집합으로 처리
Copy	Server	단말의 데이터를 복사 또는 이동
Delete	Server	권한이 충분할 때 하나의 노드를 삭제
Exec	Server	목표 노드의 값을 실행

Get	Server	목표 노드의 값을 반환
Replace	Server	존재하는 노드의 값을 변경
Sequence (복합)	Server	Sequence에 종속된 명령어들을 순서적으로 처리
Alert	Client	Notification을 전달
Result	Client	명령에 대한 답변으로 결과를 반환

본 구현에 사용한 처리방법은 단말의 메모리 및 속도 부담을 최소화하기 위하여, 순차적 처리 우선으로 한다. 즉, SAX 방식에 가깝도록 처리하였으나, 순수한 SAX 방식과는 다르다. 그림2는 순차적 처리를 위한 파서의 동작 방식을 그림으로 표현한 것이다. 이때 주의주의가 필요한 부분이 복합 명령어의 처리인데, sequence는 순차적인 처리가 필요하며, atomic은 단위 처리가 가능하도록 하여야 한다.

우선 명령어 처리는 시작태그('<')을 찾아 어떤 명령을 확인하고 해당 명령어에 따라, 복합명령어에 대해서는 순환(Recursion)호출 방식을 사용하여 처리하고, 단일 명령은 바로 처리하도록 하였다. 이 때 복합명령 처리를 위하여 회귀적인 방법을 사용하였는데, 이는 [2]에 따르면 임베디드 시스템에 부담을 주는 방식이나, 명령의 중첩 정도가 높지 않으므로 일반적으로 큰 부담을 주지 않는다.

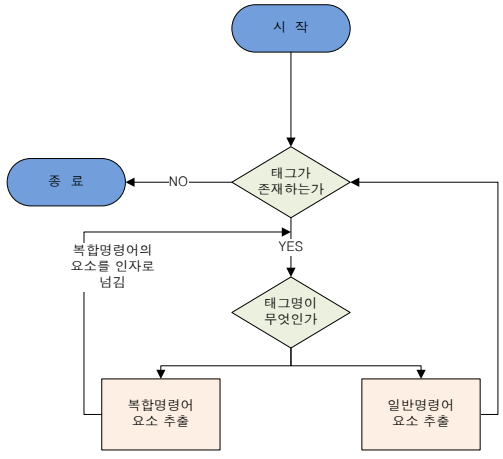

그림 2. XML Parser의 처리 형태

기존 방식들이 명령어를 찾기 위해 연속적인 비교를 함으로 인하여 복합 명령어일 경우 비교하는 횟수는 큰 폭으로 증가는 반면에 본 방식은 단 한 번만의 검색으로 모든 처리를 하므로 큰 사이즈의 XML을 빠를 시간에 처리할 수 있다.

4. 타 공개 소프트웨어와의 비교

기존에 개발된 공개 OMA-DM 소프트웨어는 syncML Toolkit과 Funambol이 대표적이다. OMA SPEC에 필요한 기능들이 비교적 잘 설계되어 있으나, 기능 상제약과 성능상의 문제를 가지고 있다. 또한 SyncML과 OMA-DM 이 별도의 프로그램으로 분리되어 있으며, 최근에는 OMA-DM은 업데이트가 안되는 것으로 보인다.

특히 XML파서에서는 본 논문의 설계방식과는 달리 하나의 메시지 안에 있는 같은 모든 명령어를 한꺼번에 추출하기 때문에, 순서적인 처리를 요구하는 복합명령 등에서 문제를 발생시킨다. 그림3은 복합명령어인 <Sequence>명령의 내부인자를 추출하면서 생길 수 있는 문제를 보여준다.

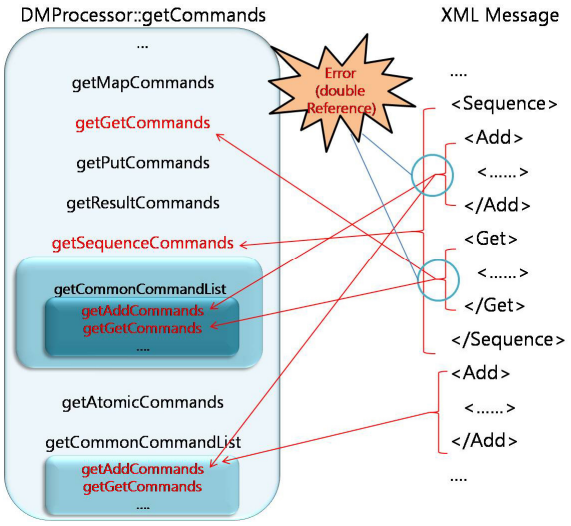

그림 3. Funambol Parser의 오동작 사례

5. 실험 결과

본 연구에서는 네비게이터와 같은 임베디드단말에서 동작할 수 있는 단말용 소프트웨어를 설계/구현 하여 Marvell 320 PDA (64MB, windows mobile 6.0) 에서 구동하였다. 향후 향후 8051/AVR 등 64Kbyte 이하의 메모리를 갖는 8Bit CPU등에 성능실험을 진행할 예정이다.

우선, 프로토콜 구현의 정합성을 확인하기 위하여 SCTS (SyncML Conformance Test Suite) 서버와 연동하여 실험하였으며, 그 결과 모든 시험을 통과할 수 있음을 확인하였다. 표 2는 특히 타 공개 소프트웨어에서 지원이 잘되지 않는 복합 명령의 경우를 비고에 표시하였다.

또한 본 연구를 공동 수행하고 있는 (주) DKI의 상용서버와 연동 시험 실시하고 있다.

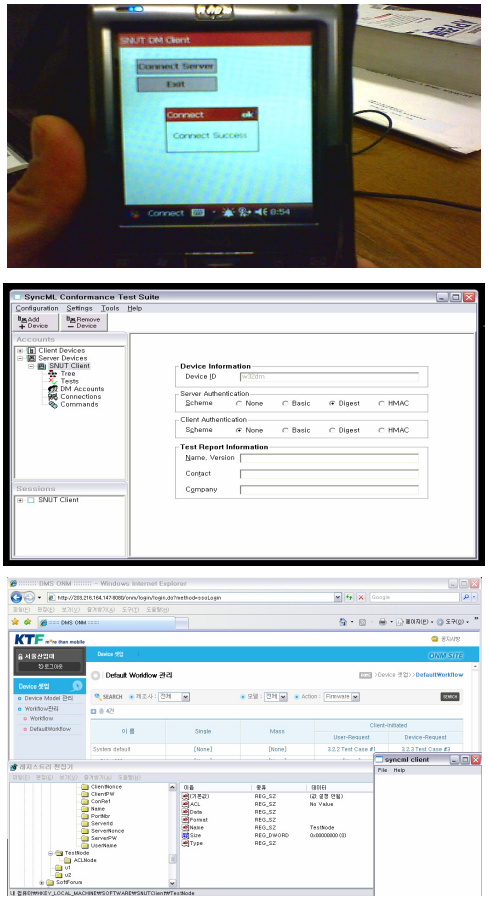

그림4. SCTS와 연동 실험

(상:테스트단말, 중: SCTS, 하: DKI DM관리서버)

6. 결 론

본 연구는 매우 제한적인 시스템환경에서 SyncML 과 OMA-DM의 XML 파서 기능을 지원하기 위한 효과적인 임베디드 파서의 구현 예를 제시하였다. OMA-DM의 XML 프로토콜 이 갖는 <Sequence>, <Atomic> 등 구조적인 처리를 요하는 구조를 반영하여 설계되었으며, 기존의 공개 소프트웨어들에서 처리하지 못하는 부분까지 처리하여 인증에 필요한 SCTS 테스트를 통과하였다. 성능면에서는 메모리 요구사항과 속도 등에 대해서는 좀더 데이터 수집이 필요하다.

표 2. SCTS 테스트항목 요약

번호	테스트요약	비고
#1	Basic 인증, Device 정보 전송	
#2	인증 재요청(basic -> md5)	
#3	Md5 인증, Get명령어 처리	○
#4	HMAC 인증	
#5	Add명령어 처리	○
#6	Replace명령어 처리	○
#7	Sequence명령어 처리	○
#8	End-User와의 interaction 처리	
#9	Delete명령어 처리	○
#10	Multiple Message 처리	
#11	Atomic명령어 처리	○
#12	Device 내 Standard-Object 구조 확인	
#13	ACL 속성의 다양한 처리	
#14	ACL 권한 변경에 의한 명령어 처리	
#15	ACL 권한, Tree구조관계에 관한 처리	
#16	Tree구조관계이해를 바탕으로 Delete명령어 처리	
#17	Large-Object 처리	
#18	Get명령어 Struct속성 처리	
#19	Get명령어 StructData속성 처리	
#20	Device Notification 처리	

감사의글

본 연구는 중소기업청 기업부설연구소 업그레이드지원사업(휴대단말기적용 DMS(Device Management System 연구개발), 과제번호: 산기업 08-1-02)의 지원을 받았음.

참고문헌

[1]Open Mobile Alliance (OMA), www.openmobilealliance.org

[2] E. M. Collado et. al., “Embedded XML DOM Parser: An Approach for XML Data Processing on Networked Embedded Systems with Real-Time Requirements,” EURASIP Journal on Embedded Systems, vol. 2008.

[3]SyncML Toolkit, sourceforge.net/projects/syncml-ctoolkit

[4]Funambol, http://www.funambol.com

[5]박주건, 박기현, 장대진, 장명숙, “OMA DM을 기반으로 한 무선 이동통신 단말기 관리 에이젼트 설계 및 구현,” 정보과학회 논문지, 컴퓨팅의 실제, 2008, 6.